UDC: 004.93 https://doi.org/10.15407/jai2025.04.100

I. Yefanov[1], **N. Shapoval**[2]
[1,2]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
37 Beresteysky Avenue, Kyiv, 03056
[1]efanov.illiya@lll.kpi.ua
[2]shovgun@gmail.com
[1]https://orcid.org/0009-0001-5736-0811
[2]https://orcid.org/0000-0002-5642-3753

PRUNING OF CONVOLUTIONAL NEURAL NETWORKS USING INTERPRETABILITY OF KOLMOGOROV-ARNOLD NETWORKS

Abstract. The paper addresses the pressing scientific and technical problem of over-parameterization in deep Convolutional Neural Networks (CNNs), which hinders their effective deployment in Edge AI systems. Traditional compression methods, such as magnitude-based pruning, often remove functionally important components as they fail to account for the semantic contribution of filters to the final decision. This study proposes a novel structural pruning method leveraging the interpretability of Kolmogorov-Arnold Networks (KAN). A hybrid CNN-KAN architecture is developed, where the KAN layer acts as an "interpretable bottleneck," enabling the analysis of convolutional feature importance through learned B-spline coefficients. A mathematical importance criterion based on the maximum weighted L2-norm of spline coefficients is formalized. An iterative pruning algorithm with adaptive fine-tuning is developed. Experimental studies on the CIFAR-10 dataset demonstrate that the proposed method achieves a compression ratio of 1.33× (reducing parameters from 11.2 million to 8.4 million) while maintaining an accuracy of 90.68%. This result outperforms classical pruning by 1.56% and is competitive with knowledge distillation methods.

Keywords: neural networks, pruning, Kolmogorov-Arnold Networks, KAN, model compression, interpretability, B-splines, Edge AI.

Introduction

The evolution of modern artificial intelligence models has led to the creation of extraordinarily powerful yet cumbersome and computationally expensive architectures. Models containing billions of parameters demonstrate impressive results in image processing and natural language tasks, but their use is often limited to powerful server systems. This creates a significant gap between the capabilities of state-of-the-art models and their practical application on resource-constrained devices. Parallel to the development of large models, there is growing demand for deploying AI directly on edge devices (Edge AI) — smartphones, IoT devices, autonomous vehicles, medical equipment. Such applications face strict constraints: limited computational power (often less than 1 TFLOPS), small memory capacity (several hundred megabytes), limited battery life, and real-time requirements (latency less than 100 ms). In response to this challenge, the field of neural network compression and optimization is actively developing. Methods such as quantization, knowledge distillation, and pruning [4] allow for significant model size reduction and acceleration. Traditional pruning methods are based on heuristic criteria, the most common being weight magnitude (Magnitude-based pruning). The main assumption is that weights with smaller absolute values make less contribution to the network output and can therefore be removed with minimal accuracy loss. However, this assumption is essentially "blind" — it does not account for the functional role of a neuron in the context of the entire network, its interaction with other components, or the semantic meaning of the features it encodes. Recently introduced Kolmogorov-Arnold Networks (KAN) architecture [1] opens new possibilities in this direction. Unlike traditional Multi-Layer Perceptrons (MLPs), where nonlinear transformations are performed by fixed activation functions on neurons, KAN has learnable activation functions on connections (edges) of the network. This makes KAN much more interpretable — each connection can be visualized as a function graph, its behavior understood, and even replaced with a symbolic mathematical formula.

Problem Statement

The main problem of existing structural pruning methods is the lack of a reliable criterion for assessing the importance of convolutional filters. Using L1 or L2 norm of filter weights as a proxy for its importance often leads to erroneous removal of filters that have small weight amplitudes but are critical for detecting rare features or correct operation of subsequent layers. Formally, let $F = F^1, F^2, \dots, Fc$ be the set of convolutional filters in a target layer, where each filter $F_i \in \mathbb{R}^{(C_{in} \times K_h \times Kw)}$. The magnitude-based importance is typically defined as:

$$I_{mag}(F_i) = |F_i|_p = \left(\sum_{c,h,w} |w_{i,c,h,w}|^p \right)^{1/p} \quad (1)$$

where p ∈ {1, 2}. This metric assumes monotonic relationship between parameter magnitude and functional importance, which is often violated in practice. Other approaches, such as gradient or Hessian-based pruning (Optimal Brain Damage, Optimal Brain Surgeon) [7], provide higher accuracy but require significant computational resources. The second-order approximation of loss change when removing weight w_i is:

$$\Delta L \approx \frac{\partial L}{\partial w_i} \Delta w_i + \frac{1}{2} \Delta w_i^T H_{ii} \Delta w_i \quad (2)$$

where H_{ii} is the diagonal element of the Hessian matrix. Computing the full Hessian has $O(n^2)$ complexity, making it impractical for deep networks.

The research problem is formulated as developing a structural pruning method for convolutional neural networks that combines computational efficiency with semantic analysis depth. To this end, we propose using the KAN architecture as an interpretable classifier layer that allows evaluating the importance of input features through analysis of learned activation function coefficients.

Related Work

Research on neural network compression methods is an active field. Works [4, 11] laid the foundations for modern pruning, demonstrating the possibility of significant model size reduction without accuracy loss. Lottery Ticket Hypothesis. The lottery ticket hypothesis [5] formulated that any dense, randomly initialized neural network contains a sparse subnetwork that, when trained in isolation, can achieve the accuracy of the original network. Formally, let $f(x; \theta^0)$ be a randomly initialized network with parameters θ₀. There exists a binary mask $m \in 0,1^{|\theta|}$ such that:

$$\text{Acc}\big(f(x; m \odot \theta_0^*)\big) \geq \text{Acc}\big(f(x; \theta^*)\big) - \epsilon \quad (3)$$

where θ* and θ₀* are the trained parameters of the full and pruned networks respectively, and ε is a small tolerance. Knowledge Distillation. The method [6] involves training a smaller model ("student") to imitate the behavior of a large model ("teacher"). The distillation loss combines hard and soft targets:

$$\mathcal{L}_{\text{KD}} = \alpha \mathcal{L}_{\text{CE}}(y, \sigma(z_S)) + (1-\alpha)\tau^2 \text{KL}(\sigma(z_T/\tau)||\sigma(z_S/\tau)) \quad (4)$$

where z_T, z_S are teacher and student logits, τ is temperature, and α balances the losses. Kolmogorov-Arnold Networks. The KAN architecture [1, 3], inspired by the Kolmogorov-Arnold representation theorem, was introduced in 2024. Unlike MLPs based on the universal approximation theorem [16], KAN demonstrate better scaling laws and higher interpretability [2], making them a promising tool for analyzing internal representations of neural networks.

The Purpose of the Study

The objective of this work is to improve the efficiency of convolutional neural network compression by developing a structural pruning method guided by an importance criterion based on KAN layers.

To achieve this objective, the following tasks must be solved:

1. Develop a hybrid CNN-KAN architecture for feature importance assessment.

2. Formalize an algorithm for iterative structural pruning using KAN-based metrics.

Method

Kolmogorov-Arnold Representation Theorem.

The theoretical foundation of KAN lies in the Kolmogorov-Arnold representation theorem [3] (1957), which states that any continuous multivariate function can be represented as a composition of univariate functions.

Theorem (Kolmogorov-Arnold): $Let\ f: [0,1]^n \to \mathbb{R}$ be a continuous function. Then there exist continuous univariate functions $\varphi q,p: [0,1] \to \mathbb{R}$ and $\Phi q: \mathbb{R} \to \mathbb{R}$ such that:

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right) \quad (5)$$

This theorem demonstrates that multi-dimensional dependencies can be decomposed into combinations of univariate functions, with addition being the only truly multidimensional operation. However, the original functions $\varphi q,p$ are non-smooth (non-differentiable almost everywhere), making direct application impossible.

Mathematical Foundations of B-splines

KAN overcomes the non-smoothness limitation by parameterizing edge functions using B-splines [9] — piecewise polynomial functions with excellent approximation properties.

A B-spline $S(x)$ of degree k on interval [t_0, t_m] is defined through basis functions $B_{i,k}(x)$ computed recursively using the Cox-de Boor formula:

Base case (degree 0):

$$B_{i,0}(x) = \begin{cases} 1, & \text{if } t_i \le x < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

Recursive step (k > 0):

$$B_{i,k}(x) = \alpha_i^k(x) B_{i,k-1}(x) + \left(1 - \alpha_{i+1}^k(x)\right) B_{i+1,k-1}(x) \quad (7)$$

where $\alpha_i^k(x) = (x - t_i)/(t_{i+k} - t_i)$. Any smooth function can be approximated as:

$$S(x) = \sum_{i=0}^{n} c_i B_{i,k}(x) \quad (8)$$

where c_i are control coefficients. Key properties of B-splines:

– Local support: $B_{i,k}(x) = 0\ for\ x \notin [t_i, t_i{}^+{}_k{}^{+1}]$

– Non-negativity: $B_{i,k}(x) \ge 0\ forall\ x$

– Partition of unity: $\sum_i B_{i,k}(x) = 1 forall x$

– Smoothness: $B_{i,k}(x) \in C^{(k-1)}$ at non-multiple knots.

The local support property is particularly important for KAN, as it ensures that changing one coefficient c_i affects the function only locally, providing stability against catastrophic forgetting.

KAN Layer Architecture

A KAN layer transforms an input vector $x \in \mathbb{R}^{(n_{in})}$ to output $y \in \mathbb{R}^{(n_{out})}$ through:

$$y_j = \sum_{i=1}^{n_{in}} \varphi_{j,i}(x_i), \quad j = 1, \dots, n_{out} \quad (9)$$

Each edge function $\varphi j,i: \mathbb{R} \to \mathbb{R}$ is parameterized as:

$$\varphi_{j,i}(x) = w_b \cdot b(x) + w_s \cdot \sum_{l=0}^{G+k-1} c_{j,i,l} B_{l,k}(x) \quad (10)$$

where:

- $b(x) = x \cdot \sigma(x)$ is the base SiLU activation
- $w_b, w_s \in \mathbb{R}$ are learnable scaling weights
- $c_{j,i,l} \in \mathbb{R}$ are spline coefficients
- G is the grid size (number of intervals)
- k is the spline degree (typically k = 3 for cubic)

The total number of parameters in a KAN layer is:

$$\text{Params}_{KAN} = n_{in} \times n_{out} \times (G + k + 2) \quad (11)$$

For a multi-layer KAN network with architecture $[n^0, n^1, \dots, nL]$:

$$x^{(0)} \to \Phi^{(1)}x^{(1)} \to \Phi^{(2)} \dots \to \Phi^{(L)}x^{(L)} \quad (12)$$

where each layer l computes:

$$x_j^{(l)} = \sum_{i=1}^{n_{l-1}} \varphi_{j,i}^{(l)}\left(x_i^{(l-1)}\right) \quad (13)$$

Hybrid CNN-KAN Architecture

The proposed architecture combines convolutional layers for spatial feature extraction with a KAN layer for interpretable feature analysis:

1. Convolutional Backbone: A sequence of residual blocks. Each residual block Rl performs:

$$h_{l+1} = R_l(h_l) = \text{ReLU}(C_l(h_l) + h_l) \quad (14)$$

where Cl is a composition of convolution, batch normalization, and ReLU:

$$C_l(h) = \text{ReLU}\left(\text{BN}\left(\text{Conv}_l(h)\right)\right) \quad (15)$$

2. Global Average Pooling (GAP): Converts feature maps $A \in \mathbb{R}\hat{}(C \times H \times W)$ to feature vector $f \in \mathbb{R}^C$:

$$f_c = \frac{1}{H \cdot W}\sum_{h=1}^{H}\sum_{w=1}^{W} A_{c,h,w} \quad (16)$$

The GAP operation ensures spatial invariance and creates a direct correspondence between each convolutional filter and one element of the feature vector.

3. KAN Bottleneck: The feature vector is processed by a KAN layer with L latent neurons (L < C):

$$z_j = \sum_{i=1}^{C} \varphi_{j,i}(f_i), \quad j = 1, \dots, L \quad (17)$$

This creates an information bottleneck that forces the network to identify the most important features.

4. Linear Classifier: Maps latent representation to class logits:

$$y = W_{cls}z + b_{cls} \quad (18)$$

where $W_{cls} \in \mathbb{R}^{(K\times L)}$ and $b_{cls} \in \mathbb{R}^K$.

Filter Importance Criterion

The key hypothesis is that filter importance correlates with the amplitude and complexity of the KAN functions processing its output. We formalize this through the influence matrix.

Definition 1 (Influence Coefficient): The influence of input i on latent neuron j is:

$$I_{ji} = \left|w_{s,ji}\right| \cdot |c_{ji}|_2 \quad (19)$$

where $\|cji\|^2 = \sqrt{\sum_{l=0}^{G+k-1}\left(c_{ji,l}^2\right)}$ is the L2-norm of spline coefficients. This metric captures both the spline weight $w_{s,ji}$ and the L2-norm of spline coefficients. A large I_{ji} indicates that changes in f_i significantly affect z_j through a high-amplitude nonlinear transformation.

Definition 2 (Filter Importance): The importance of filter i is:

$$\text{Importance}(i) = \max_{j\in\{1,\dots,L\}} I_{ji} \quad (20)$$

The max operation is motivated by the specialization principle: if a filter strongly influences even one latent neuron, it encodes unique semantic information essential for that aspect of the decision.

Theoretical Justification: Consider the gradient of the loss with respect to feature f_i:

$$\frac{\partial \mathcal{L}}{\partial f_i} = \sum_{j=1}^{L} \frac{\partial \mathcal{L}}{\partial z_j} \cdot \varphi'_{j,i}(f_i) \quad (21)$$

For a B-spline function, the derivative satisfies:

$$\left|\varphi'_{j,i}(x)\right| \le \left|w_{s,ji}\right| \cdot \max_l\left|c_{ji,l}\right| \cdot \text{const} \quad (22)$$

Thus, Iji provides an upper bound on the sensitivity of the latent representation to changes in fi.

Iterative Pruning Algorithm

We now present the formal algorithm for KAN-guided structural pruning.

The RemoveFilters operation performs structural modification:

– For convolutional layer: $W_{new} = W_{old}[R, :, :, :]$.

– For batch normalization: $\gamma_{new} = \gamma_{old}[R], \beta_{new} = \beta_{old}[R]$.

– For KAN layer: remove corresponding input dimensions.

The FineTune operation uses Adam optimizer [14] with cosine annealing learning rate schedule:

$$\eta_t = \eta_0 \cdot \frac{1 + \cos(\pi t/T)}{2} \qquad (23)$$

where η_0 is the initial learning rate and T is the total number of steps.

Algorithm 1. KAN-Guided Iterative Pruning

Require: Trained model M_0, pruning ratio p, iterations N, fine-tune epochs E
Ensure: Pruned model M_N

$M \leftarrow M_0$
$\text{acc}_0 \leftarrow \text{Evaluate}(M, \mathcal{D}_{\text{val}})$
for $k = 1$ to N **do**
  $C \leftarrow$ number of filters in target layer
  for $i = 1$ to C **do**
    for $j = 1$ to L **do**
      $I_{ji} \leftarrow |w_{s,ji}| \cdot \sqrt{\sum_{l=0}^{G+k-1} c_{ji,l}^2}$
    end for
    $\text{Importance}(i) \leftarrow \max_j I_{ji}$
  end for
  $\sigma \leftarrow \text{argsort}(\text{Importance})$
  $n_{\text{prune}} \leftarrow \lfloor p \cdot C \rfloor$
  $\mathcal{P} \leftarrow \{\sigma_1, \ldots, \sigma_{n_{\text{prune}}}\}$
  $\mathcal{R} \leftarrow \{1, \ldots, C\} \setminus \mathcal{P}$
  $M' \leftarrow \text{RemoveFilters}(M, \mathcal{P})$
  $M \leftarrow \text{FineTune}(M', \mathcal{D}_{\text{train}}, E)$
  $\text{acc}_k \leftarrow \text{Evaluate}(M, \mathcal{D}_{\text{val}})$
  if $\text{acc}_k < \text{acc}_0 - \delta$ **then**
    break
  end if
end for
return M

Fig. 1. Algorithm pseudocode

Computational Complexity Analysis

Let C be the number of filters, L the number of latent neurons, G the spline grid size, and |D| the dataset size.

Per-iteration complexity:

– Importance computation: $O(C \cdot L \cdot G)$

– Sorting: $O(ClogC)$

– Filter removal: O(C)

– Fine-tuning: $O\left(E \cdot |D| \cdot T_{forward}\right)$

Total complexity: $O\left(N \cdot E \cdot |D| \cdot T_{forward}\right)$, dominated by fine-tuning.

Experiment

Experiments were conducted on the CIFAR-10 dataset [17] using a modified ResNet-18 architecture [8] and PyTorch framework [13].

Experimental Setup:

– Dataset: CIFAR-10 (50,000 training, 10,000 test images, 32×32 resolution, 10 classes)

– Architecture: ResNet-18 backbone with KAN bottleneck (L = 256, G = 5, k = 3)

– Training: Adam optimizer, learning rate 10^{-3}, batch size 128, 200 epochs

– Pruning: N = 5 iterations, p = 10% per iteration, E = 20 fine-tuning epochs

Three methods were compared: classical magnitude-based pruning, knowledge distillation [6], and the proposed KAN-guided method.

Table 1. Comparative accuracy and compression results

Method	Params	Acc.	ΔAcc.
Original	11.2 M	92.45%	—
Mag. Prune	8.4 M	89.12%	-3.33%
KD	8.4 M	89.85%	-2.60%
KAN-Guided	8.4 M	90.68%	-1.77%

Results show that the KAN-guided method achieves superior accuracy retention, outperforming magnitude pruning by 1.56*% and knowledge distillation by 0.83%.*

Table 2. Computational efficiency metrics

Model	FLOPs	Speedup	Latency
Original	556 M	1.00×	23.4 ms
KAN-Pruned	418 M	1.32×	17.7 ms

The 25% FLOPs reduction translates to 1.32× real-world speedup on CPU, confirming practical benefits for edge deployment.

Ablation Study: We analyzed the effect of spline grid size G on pruning quality:

$$\text{Accuracy}(G) = \begin{cases} 88.2\% & G = 3 \\ 90.68\% & G = 5 \\ 90.81\% & G = 10 \end{cases} \quad (24)$$

While G = 10 slightly improves accuracy, it increases KAN parameters by 2×. The optimal trade-off is G = 5.

Conclusions

This work successfully developed theoretical foundations and methodology for a novel approach to structural neural network pruning based on KAN interpretability. The proposed method transcends simple heuristics by providing a semantically justified criterion for network component importance. The key innovation lies in using the KAN layer not merely as a building block but as an analysis and interpretation tool. The mathematical framework based on B-spline coefficient analysis enables quantitative assessment of feature importance, while the iterative algorithm ensures gradual adaptation to reduced capacity. Experimental validation on CIFAR-10 demonstrates the method's superiority over classical magnitude-based pruning and competitive performance with knowledge distillation. The achieved 1.32× inference speedup opens prospects for real-time systems and mobile platforms. The compression ratio of 1.33× with only 1.77% accuracy loss represents a favorable trade-off for edge AI applications. Future research directions include: (1) extension to transformer architectures, (2) investigation of layer-wise importance criteria, (3) combination with quantization for further compression, and (4) theoretical analysis of pruning limits under KAN-guided criteria. The work demonstrates that architectural innovations like KAN can serve dual purposes: improving model expressiveness while simultaneously enabling more intelligent optimization and compression strategies.

References

1. Liu, Z., Wang, Y., Vaidya, S., et al. (2024). KAN: Kolmogorov-Arnold Networks. arXiv preprint arXiv:2404.19756.
2. Blealtan. (2024). efficient-kan: An efficient pure-PyTorch implementation of Kolmogorov-Arnold Network. GitHub repository. https://github.com/Blealtan/efficient-kan
3. Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. Doklady Akademii Nauk SSSR, 114(5), 953-956.
4. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural networks. In Advances in Neural Information Processing Systems (pp. 1135-1143).
5. Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In International Conference on Learning Representations.
6. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.
7. LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In Advances in Neural Information Processing Systems (pp. 598-605).
8. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).
9. De Boor, C. (1972). On calculating with B-splines. Journal of Approximation Theory, 6(1), 50-62.
10. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning (pp. 448-456).
11. Blalock, D., Ortiz, J. J. G., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning? Proceedings of Machine Learning and Systems, 2, 129-146.
12. Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge distillation: A survey. International Journal of Computer Vision, 129, 1789-1819.
13. Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems (pp. 8026-8037).
14. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR).
15. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

16. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 2(4), 303-314.

17. Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

The article has been sent to the editors 05.12.25.
After processing 10.12.25.
Submitted for printing 30.12.25.

Copyright under license CCBY-SA4.0.